# CS 225

**Data Structures**

# heap-puzzle3.cpp

```
 1 #include <iostream>
 2 using namespace std;
 3
 4 int main() {
 5   int *x = new int;
 6   int &y = *x;
 7
 8   y = 4;
 9
10   cout << &x << endl;
11   cout << x << endl;
12   cout << *x << endl;
13
14   cout << &y << endl;
15   cout << y << endl;
16   cout << *y << endl;
17 }
```

# joinSpheres-byValue.cpp

```
11 /*
12  * Creates a new sphere that contains the exact volume
13  * of the volume of the two input spheres.
14  */
15 Sphere joinSpheres(Sphere s1, Sphere s2) {
16   double totalVolume = s1.getVolume() + s2.getVolume();
17
18   double newRadius = std::pow(
19     (3.0 * totalVolume) / (4.0 * 3.141592654),
20     1.0/3.0
21   );
22
23   Sphere result(newRadius);
24
25   return result;
26 }
```

```
28 int main() {
29   Sphere *s1 = new Sphere(4);
30   Sphere *s2 = new Sphere(5);
31
32   Sphere s3 = joinSpheres(*s1, *s2);
33
34   return 0;
35 }
```

# joinSpheres-byPointer.cpp

```
11 /*
12  * Creates a new sphere that contains the exact volume
13  * of the volume of the two input spheres.
14  */
15 Sphere joinSpheres(Sphere *s1, Sphere *s2) {
16   double totalVolume = s1->getVolume() + s2->getVolume();
17
18   double newRadius = std::pow(
19     (3.0 * totalVolume) / (4.0 * 3.141592654),
20     1.0/3.0
21   );
22
23   Sphere result(newRadius);
24
25   return result;
26 }
```

```
28 int main() {
29   Sphere *s1 = new Sphere(4);
30   Sphere *s2 = new Sphere(5);
31
32   Sphere s3 = joinSpheres(s1, s2);
33
34   return 0;
35 }
```

# joinSpheres-byReference.cpp

```
11 /*
12  * Creates a new sphere that contains the exact volume
13  * of the volume of the two input spheres.
14  */
15 Sphere joinSpheres(Sphere &s1, Sphere &s2) {
16   double totalVolume = s1.getVolume() + s2.getVolume();
17
18   double newRadius = std::pow(
19     (3.0 * totalVolume) / (4.0 * 3.141592654),
20     1.0/3.0
21   );
22
23   Sphere result(newRadius);
24
25   return result;
26 }
```

```
28 int main() {
29   Sphere *s1 = new Sphere(4);
30   Sphere *s2 = new Sphere(5);
31
32   Sphere s3 = joinSpheres(*s1, *s2);
33
34   return 0;
35 }
```

# Parameter Passing Properties

| | **By Value**<br>`void foo(Sphere a) { … }` | **By Pointer**<br>`void foo(Sphere *a) { … }` | **By Reference**<br>`void foo(Sphere &a) { … }` |
|---|---|---|---|
| Exactly what is copied when the function is invoked? | | | |
| Does modification of the passed in object modify the caller's object? | | | |
| Is there always a valid object passed in to the function? | | | |
| Speed | | | |
| Programming Safety | | | |

# const Parameters

# joinSpheres-byValue-const.cpp

```
11 /*
12  * Creates a new sphere that contains the exact volume
13  * of the volume of the two input spheres.
14  */
15 Sphere joinSpheres(const Sphere s1, const Sphere s2) {
16   double totalVolume = s1.getVolume() + s2.getVolume();
17
18   double newRadius = std::pow(
19     (3.0 * totalVolume) / (4.0 * 3.141592654),
20     1.0/3.0
21   );
22
23   Sphere result(newRadius);
24
25   return result;
26 }
```

```
28 int main() {
29   Sphere *s1 = new Sphere(4);
30   Sphere *s2 = new Sphere(5);
31
32   Sphere s3 = joinSpheres(*s1, *s2);
33
34   delete s1; s1 = NULL;
35   delete s2; s2 = NULL;
36
37   return 0;
28 }
```

# joinSpheres-byPointer-const.cpp

```
11 /*
12  * Creates a new sphere that contains the exact volume
13  * of the volume of the two input spheres.
14  */
15 Sphere joinSpheres(Sphere const *s1, Sphere const *s2) {
16   double totalVolume = s1->getVolume() + s2->getVolume();
17
18   double newRadius = std::pow(
19     (3.0 * totalVolume) / (4.0 * 3.141592654),
20     1.0/3.0
21   );
22
23   Sphere result(newRadius);
24
25   return result;
26 }
```

```
28 int main() {
29   Sphere *s1 = new Sphere(4);
30   Sphere *s2 = new Sphere(5);
31
32   Sphere s3 = joinSpheres(s1, s2);
33
34   delete s1; s1 = NULL;
35   delete s2; s2 = NULL;
36
37   return 0;
28 }
```

# joinSpheres-byReference-const.cpp

```
11 /*
12  * Creates a new sphere that contains the exact volume
13  * of the volume of the two input spheres.
14  */
15 Sphere joinSpheres(const Sphere &s1, const Sphere &s2) {
16   double totalVolume = s1.getVolume() + s2.getVolume();
17
18   double newRadius = std::pow(
19     (3.0 * totalVolume) / (4.0 * 3.141592654),
20     1.0/3.0
21   );
22
23   Sphere result(newRadius);
24
25   return result;
26 }
```

```
28 int main() {
29   Sphere *s1 = new Sphere(4);
30   Sphere *s2 = new Sphere(5);
31
32   Sphere s3 = joinSpheres(*s1, *s2);
33
34   delete s1; s1 = NULL;
35   delete s2; s2 = NULL;
36
37   return 0;
28 }
```

```
[waf@linux-a2 5]$ clang++ -fno-elide-constructors -std=c++11 -stdlib=libc++ -O0
joinSpheres-byValue-const.cpp sphere.cpp
joinSpheres-byValue-const.cpp:16:24: error: member function 'getVolume' not
      viable: 'this' argument has type 'const cs225::Sphere', but function is
      not marked const
  double totalVolume = s1.getVolume() + s2.getVolume();
                       ^~
./sphere.h:12:12: note: 'getVolume' declared here
    double getVolume() ;
           ^
joinSpheres-byValue-const.cpp:16:41: error: member function 'getVolume' not
      viable: 'this' argument has type 'const cs225::Sphere', but function is
      not marked const
  double totalVolume = s1.getVolume() + s2.getVolume();
                                        ^~
./sphere.h:12:12: note: 'getVolume' declared here
    double getVolume() ;
           ^
2 errors generated.
```

# const functions in classes

## sphere.h

```
1  #ifndef SPHERE_H
2  #define SPHERE_H
3
4  namespace cs225 {
5    class Sphere {
6      public:
7        Sphere();
8        Sphere(double r);
9
10       double getRadius();
11       double getVolume();
12
13       void setRadius(double r);
14
15     private:
16       double r_;
17
18   };
19 }
20
21 #endif
```

## sphere.cpp

```
1  #include "sphere.h"
2
3  namespace cs225 {
4    Sphere::Sphere() : Sphere(1) { }
5
6    Sphere::Sphere(double r) {
7       r_ = r;
8    }
9
10   double Sphere::getRadius() {
11     return r_;
12   }
13
14   double Sphere::getVolume() {
15     return (4 * r_ * r_ * r_ *
16             3.14159265) / 3.0;
17   }
18
19   void setRadius(double r) {
20      r_ = r;
21   }
22 }
```

# Copy Constructor

# Copy Constructor

**Automatic Copy Constructor**

**Custom Copy Constructor**

## sphere.h

```
 1  #ifndef SPHERE_H
 2  #define SPHERE_H
 3
 4  namespace cs225 {
 5    class Sphere {
 6      public:
 7        Sphere();
 8        Sphere(double r);
 9
10
11        double getRadius() const;
12        double getVolume() const;
13
14        void setRadius(double r);
15
16      private:
17        double r_;
18
19    };
20  }
21
22  #endif
```

## sphere.cpp

```
 1  #include "sphere.h"
 2  #include <iostream>
 3
 4  using namespace std;
 5
 6  namespace cs225 {
 7    Sphere::Sphere() : Sphere(1) {
 8       cout << "Default ctor" << endl;
 9    }
10
11    Sphere::Sphere(double r) {
12      cout << "1-param ctor" << endl;
13      r_ = r;
14    }
15
16
17
18
19
20
21
…     // ...
```

# joinSpheres-byValue-const.cpp

```
11 /*
12  * Creates a new sphere that contains the exact volume
13  * of the volume of the two input spheres.
14  */
15 Sphere joinSpheres(const Sphere s1, const Sphere s2) {
16   double totalVolume = s1.getVolume() + s2.getVolume();
17
18   double newRadius = std::pow(
19     (3.0 * totalVolume) / (4.0 * 3.141592654),
20     1.0/3.0
21   );
22
23   Sphere result(newRadius);
24
25   return result;
26 }
```

```
28 int main() {
29   Sphere *s1 = new Sphere(4);
30   Sphere *s2 = new Sphere(5);
31
32   Sphere s3 = joinSpheres(*s1, *s2);
33
34   delete s1; s1 = NULL;
35   delete s2; s2 = NULL;
36
37   return 0;
28 }
```

# Calls to constructors

| | By Value<br>`void foo(Sphere a) { … }` | By Pointer<br>`void foo(Sphere *a) { … }` | By Reference<br>`void foo(Sphere &a) { … }` |
|---|---|---|---|
| `Sphere::Sphere()` | | | |
| `Sphere::Sphere(double)` | | | |
| `Sphere::Sphere(const Sphere&)` | | | |

# joinSpheres-byPointer-const.cpp

```
11 /*
12  * Creates a new sphere that contains the exact volume
13  * of the volume of the two input spheres.
14  */
15 Sphere joinSpheres(Sphere const *s1, Sphere const *s2) {
16   double totalVolume = s1->getVolume() + s2->getVolume();
17
18   double newRadius = std::pow(
19     (3.0 * totalVolume) / (4.0 * 3.141592654),
20     1.0/3.0
21   );
22
23   Sphere result(newRadius);
24
25   return result;
26 }
```

```
28 int main() {
29   Sphere *s1 = new Sphere(4);
30   Sphere *s2 = new Sphere(5);
31
32   Sphere s3 = joinSpheres(s1, s2);
33
34   delete s1; s1 = NULL;
35   delete s2; s2 = NULL;
36
37   return 0;
28 }
```

# joinSpheres-byReference-const.cpp

```
11 /*
12  * Creates a new sphere that contains the exact volume
13  * of the volume of the two input spheres.
14  */
15 Sphere joinSpheres(const Sphere &s1, const Sphere &s2) {
16   double totalVolume = s1.getVolume() + s2.getVolume();
17
18   double newRadius = std::pow(
19     (3.0 * totalVolume) / (4.0 * 3.141592654),
20     1.0/3.0
21   );
22
23   Sphere result(newRadius);
24
25   return result;
26 }
```

```
28 int main() {
29   Sphere *s1 = new Sphere(4);
30   Sphere *s2 = new Sphere(5);
31
32   Sphere s3 = joinSpheres(*s1, *s2);
33
34   delete s1; s1 = NULL;
35   delete s2; s2 = NULL;
36
37   return 0;
28 }
```

# CS 225 – Things To Be Doing

## Register for Exam 1 (CBTF)

**More Info:** https://courses.engr.illinois.edu/cs225/fa2017/exams/

## Complete lab_debug

Due on Sunday, Sept 10th (11:59pm)

## Finish MP1 – Due Monday

Due on Monday, Sept 11th (11:59pm)

*MP2 Release: Sept 12th – Up to +7 Extra Credit for Early Submission*

## POTD

Every Monday-Friday – *Worth +1 Extra Credit /problem (up to +40 total)*